

CSCI 3753: Operating Systems

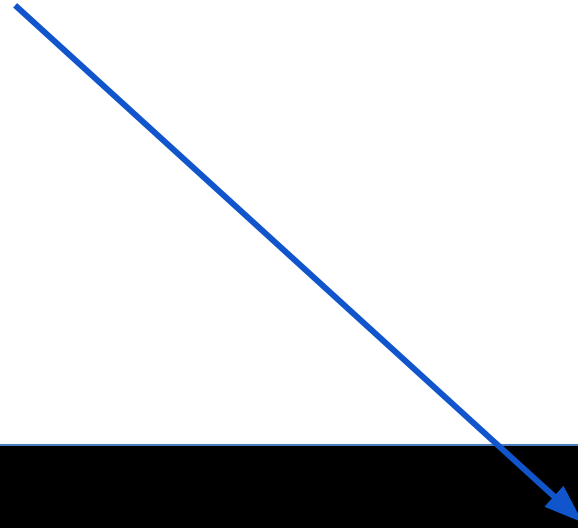
Week 5
September 26, 2025






University of Colorado **Boulder**

Announcements

- PA3 due Sunday at midnight!
- Quiz 5 due today at midnight
- Interview grading slots are now available on Canvas
 - Sign up BEFORE Tuesday
 - As the interviews get closer, the slot availability will lessen!
- As always recitation materials are stored here



Recap

- PA0,1,2 
- Problem Set 1 
- PA3 
- Introduced:
 - C stdin + stdout



PA 3

- create a Device Driver Module (LKM)
- implement file operations
 - open, seek, read, write, release
- make and load the module
- create a Device File for this Device



PA 3



- create a Device Driver Module (LKM)
 - implement file operations
 - open, seek, read, write, release
 - make and load the module
 - create a Device File for this Device
-
- **create a test program**

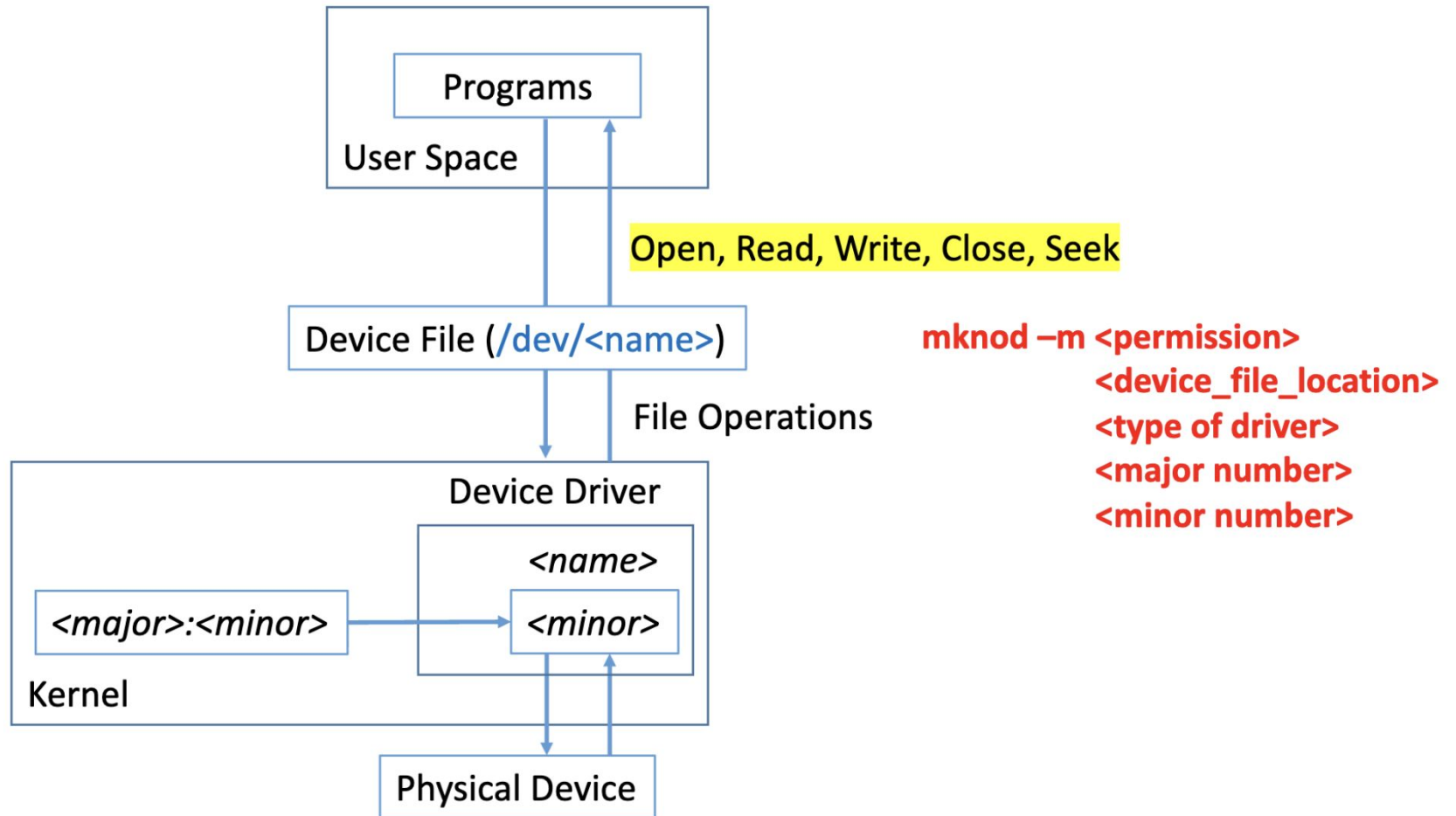
PA 2



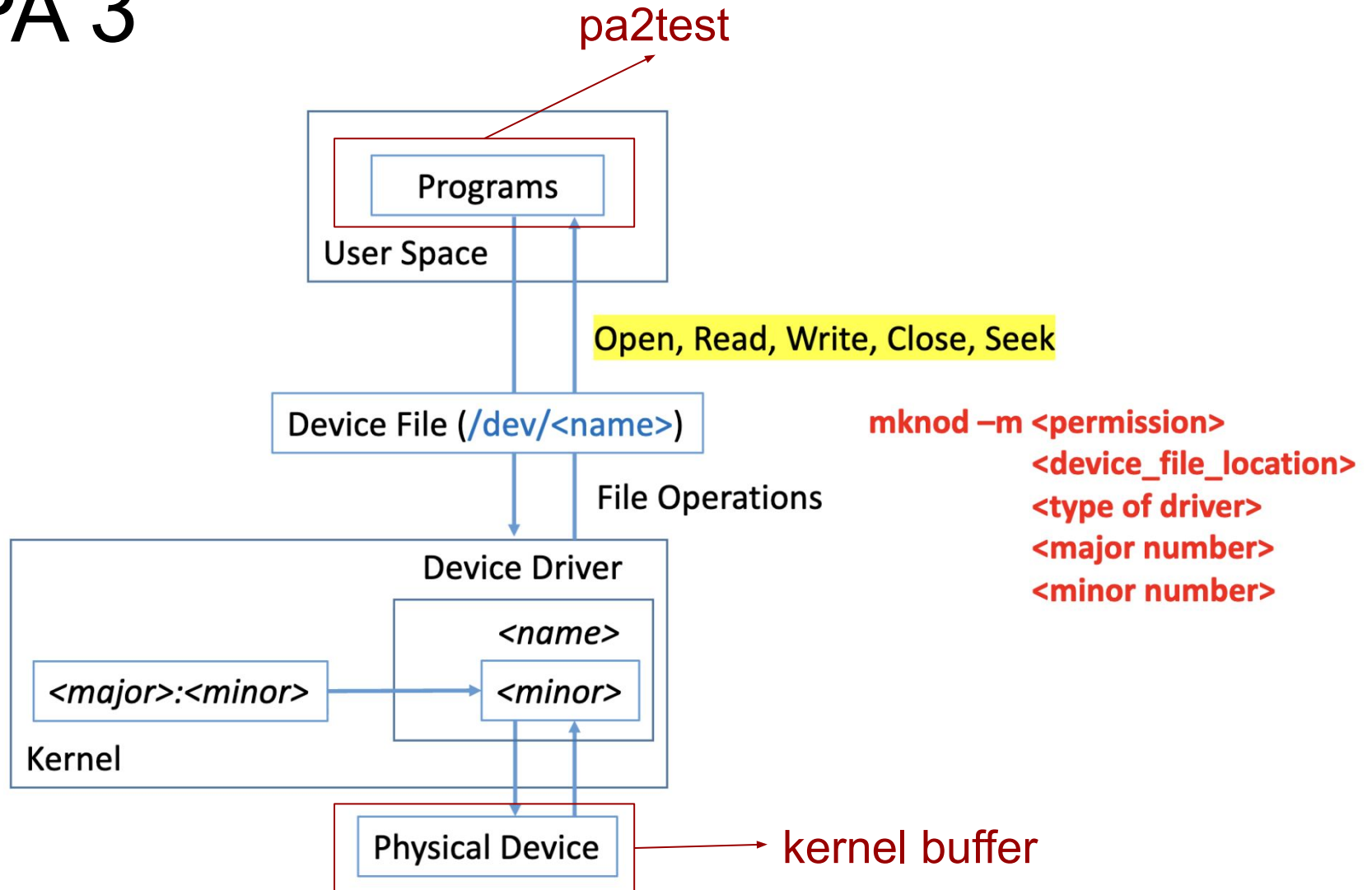
test with: `./pa2test /dev/pa3_char_device`



PA 3

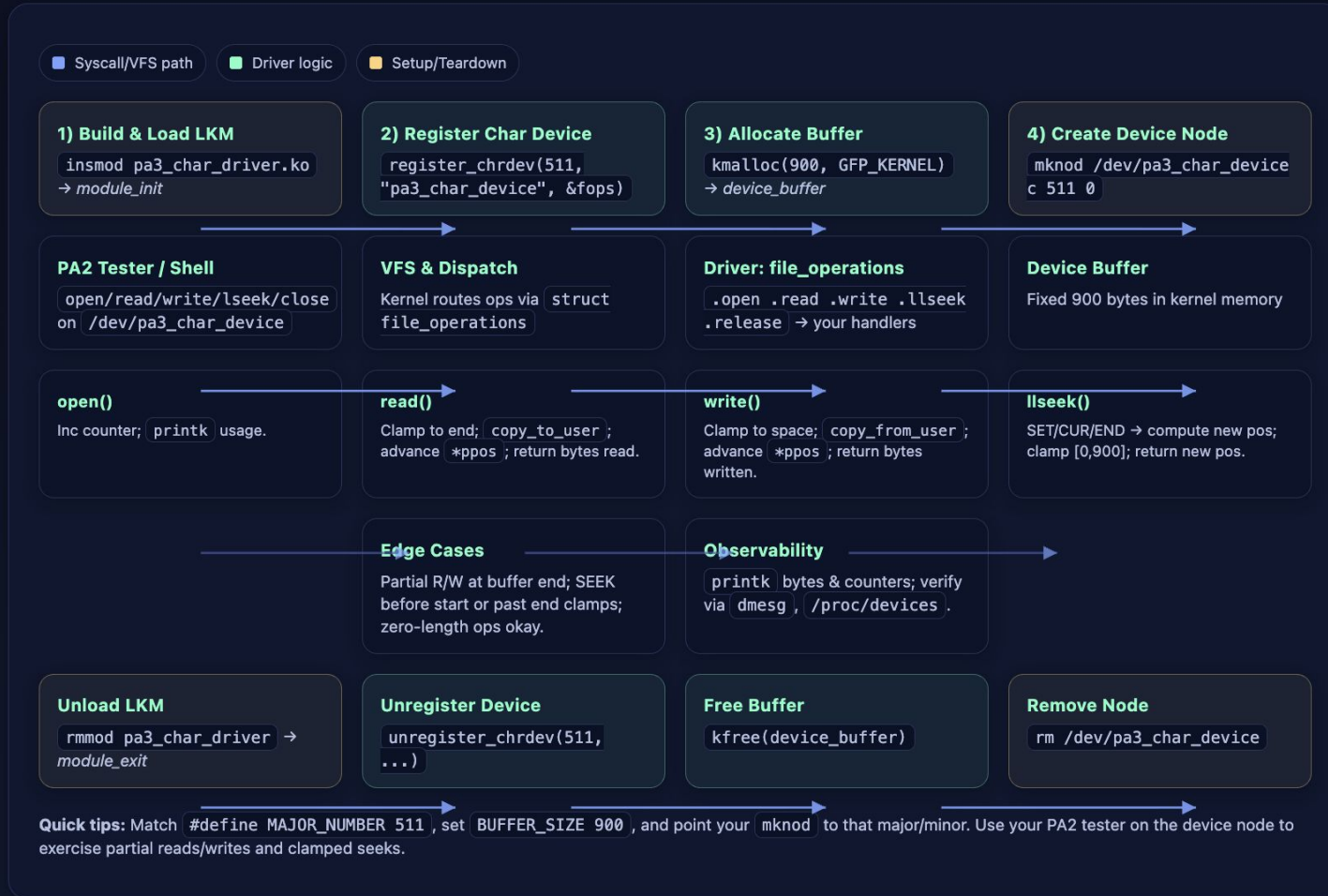


PA 3



PA3 – Character Device Driver (LKM) Flowchart

High-level data/control flow for a simple character driver that stores bytes in a fixed in-kernel buffer and exposes `open/read/write/lseek/release` via `file_operations`. Use this with the PA2 tester and a device node in `/dev`.



PA3 Demo

- Available on Canvas
- Be sure to install it on your generic kernel(5.15.0-118-generic)



PA3 - Functionality

- `module_init`
- `module_exit`
- `open`
- `release`
- `read`
- `write`
- `lseek`



PA3 - Functionality

- `module_init`
 - Announce the module is loading
 - Allocate the kernel buffer
 - Register the char device with the kernel
 - If anything fails, clean up memory and print error
- `module_exit`
- `open`
- `release`
- `read`
- `write`
- `llseek`



PA3 - Functionality

- `module_init`
- `module_exit`
 - Announce the module is unloading
 - Deregister the char device
 - Free the kernel memory
- `open`
- `release`
- `read`
- `write`
- `llseek`



PA3 - Functionality

- module_init
- module_exit
- open
 - Increment open counter
 - log open count
- release
- read
- write
- llseek



PA3 - Functionality

- module_init
- module_exit
- open
- **release**
 - Increment close counter
 - log close count
- read
- write
- lseek



PA3 - Functionality

- `module_init`
- `module_exit`
- `open`
- `release`
- `read`
 - Look at the caller's current file position
 - If at end of buffer, indicate there is nothing left to read
 - Otherwise, determine how many bytes are actually available
 - Copy that many bytes into the caller's buffer in user space
 - Advance the caller's file position by the number of bytes delivered
 - Report how many bytes were delivered, or report error
- `write`
- `lseek`



PA3 - Functionality

- module_init
- module_exit
- open
- release
- read
- write
 - Look at the caller's current position
 - If at end of buffer, indicate there is no room left
 - Otherwise, determine how much space remains
 - Copy that many bytes from the caller's user space buffer into the device buffer
 - Advance the caller's position by number of bytes stored
 - Report how many bytes were stored, or error
- lseek



PA3 - Functionality

- module_init
- module_exit
- open
- release
- read
- write
- lseek
 - Depending on reference point:
 - Set the new position to an absolute value from start
 - or add an offset to the current position
 - or add an offset to the end of the buffer
 - Clamp the resulting position
 - Update the file position and report back
 - Seek should **never** fail



PA3 - Installation

- Available on Canvas



PA3 - Installation

1. Write LKM code
2. Add module to Makefile
 - a. `obj-m:=pa3_char_driver.o`
3. Compile the module to get the .o file
 - a. `make -C /lib/modules/$(uname -r)/build M=$PWD`
4. Insert the mod into the running kernel
 - a. `sudo insmod pa3_char_driver.ko`
5. Create device node
 - a. `sudo mknod /dev/pa3_char_device c 511 0`
 - b. `sudo chmod 666 /dev/pa3_char_device`

Same as PA1



PA3 - Deinstallation

- Remove node:
 - `sudo rmmod pa3_char_driver`
 - `sudo rm -f /dev/pa3_char_device`



PA3 - Testing

- Use PA2
 - `./pa2test /dev/pa3_char_device`



Looking Ahead

- PA4
 - Create a shared array
- PA5
 - Create a DNS resolver
- PA6
 - Combine these to create a multi-threaded DNS resolver



Looking Ahead

- PA4
 - Create a shared array
- PA5
 - Create a DNS resolver
- PA6
 - Combine these to create a multi-threaded DNS resolver

Bounded Buffer Problem!!



Looking Ahead

- PA4
 - Create a shared array
 - Must be:
 - thread safe!
 - built on top of one (or more) contiguous, linear memory arrays
 - Could be:
 - FIFO
 - LIFO
 - Circular queue
 - Cannot have:
 - linked lists, dictionaries, trees, other pre-built data structures in C
 - Turn in both .h (header) and .c files for your shared array



Circular Queue

01
Step

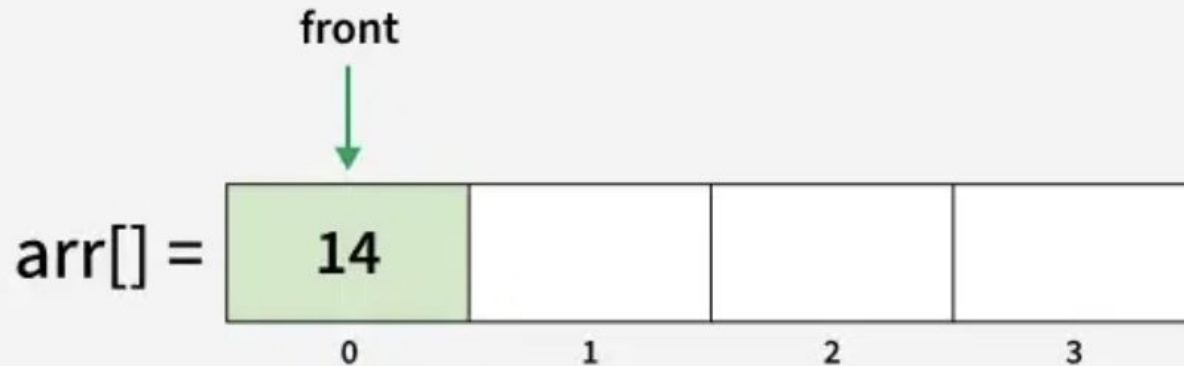
Enqueue element 14

Initially $\text{front} = 0$, $\text{size} = 0$, $\text{capacity} = 4$

$\text{rear} = (\text{front} + \text{size}) \% \text{capacity} = (0 + 0) \% 4 = 0$

$\text{arr}[0] = 14$.

$\text{size} = \text{size} + 1$



Operations in Circular Queue



Circular Queue

02
Step

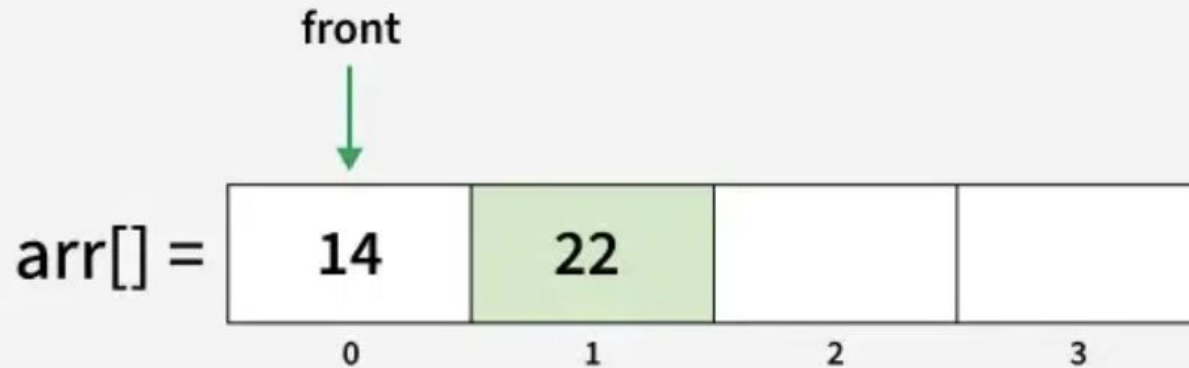
Enqueue element 22

Now front = 0, size = 1, capacity = 4

$\text{rear} = (\text{front} + \text{size}) \% \text{capacity} = (0 + 1) \% 4 = 1$

$\text{arr}[1] = 22.$

size = size + 1



Operations in Circular Queue



Circular Queue

03
Step

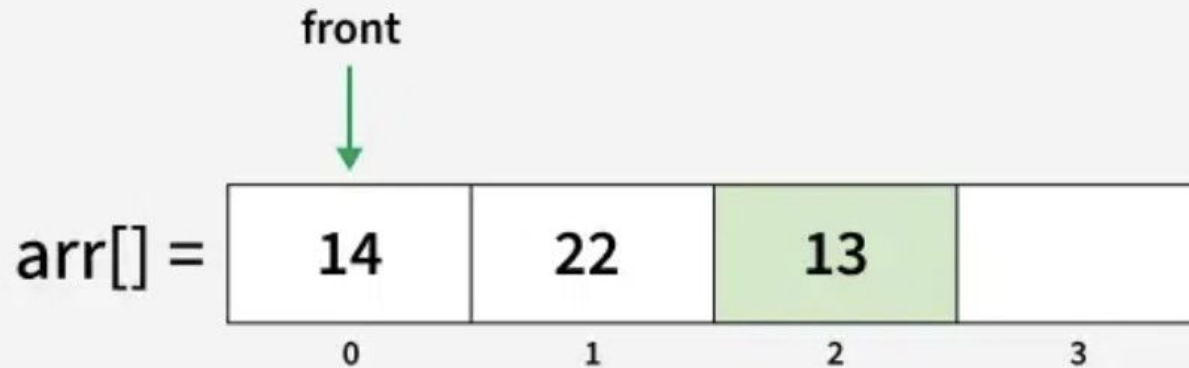
Enqueue element 13

Now front = 0, size = 2, capacity = 4

$\text{rear} = (\text{front} + \text{size}) \% \text{capacity} = (0 + 2) \% 4 = 2$

$\text{arr}[2] = 13.$

size = size + 1



Operations in Circular Queue



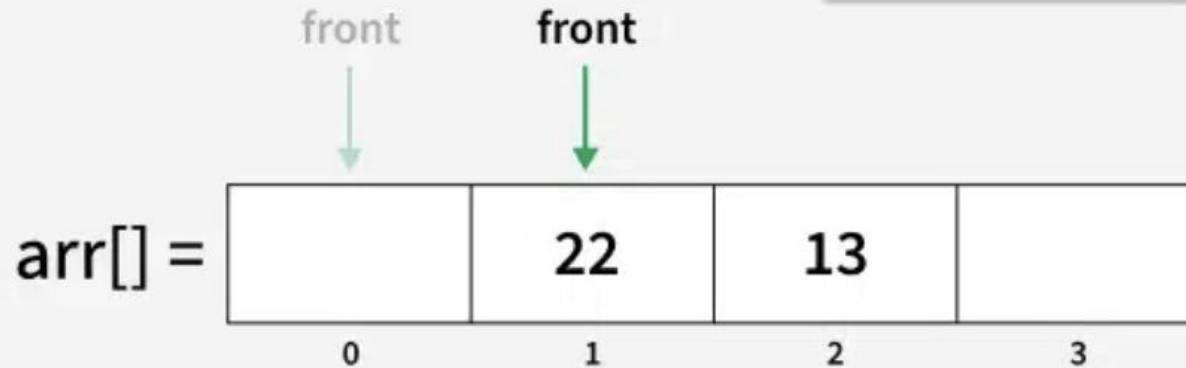
Circular Queue

04
Step

Dequeue

Now front = 0, size = 3, capacity = 4.

$\text{front} = (\text{front} + 1) \% \text{capacity}$
 $\text{size} = \text{size} - 1$



Operations in Circular Queue



Circular Queue

05
Step

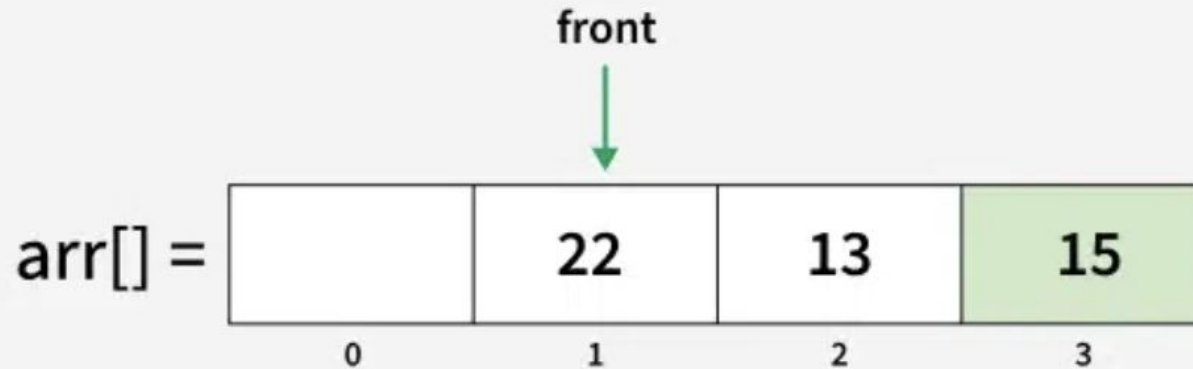
Enqueue element 15

Now front = 1, size = 2, capacity = 4

$\text{rear} = (\text{front} + \text{size}) \% \text{capacity} = (1 + 2) \% 4 = 3$

$\text{arr}[3] = 15.$

size = size + 1



Operations in Circular Queue



Circular Queue

06
Step

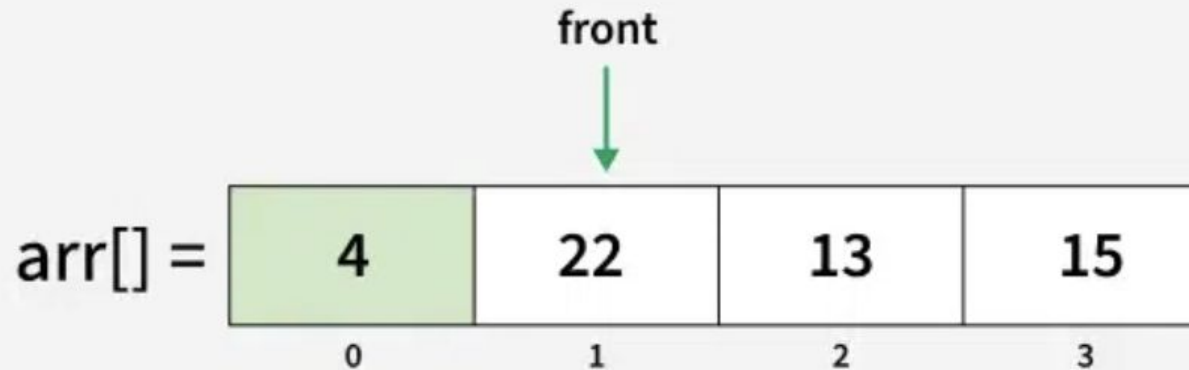
Enqueue element 4

Now front = 1, size = 3, capacity = 4

$\text{rear} = (\text{front} + \text{size}) \% \text{capacity} = (1 + 3) \% 4 = 0$

$\text{arr}[0] = 4.$

size = size + 1



Operations in Circular Queue



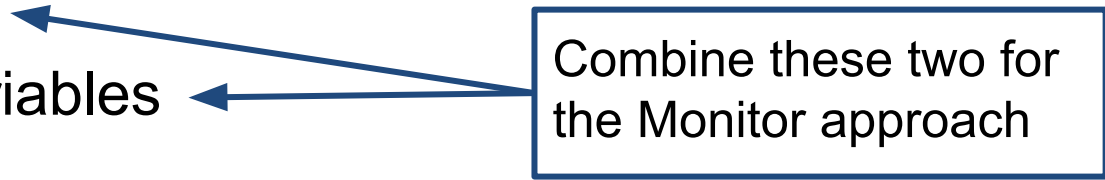
Thread Safe Options

- Mutex Locks
- Condition variables
- Semaphores



Thread Safe Options

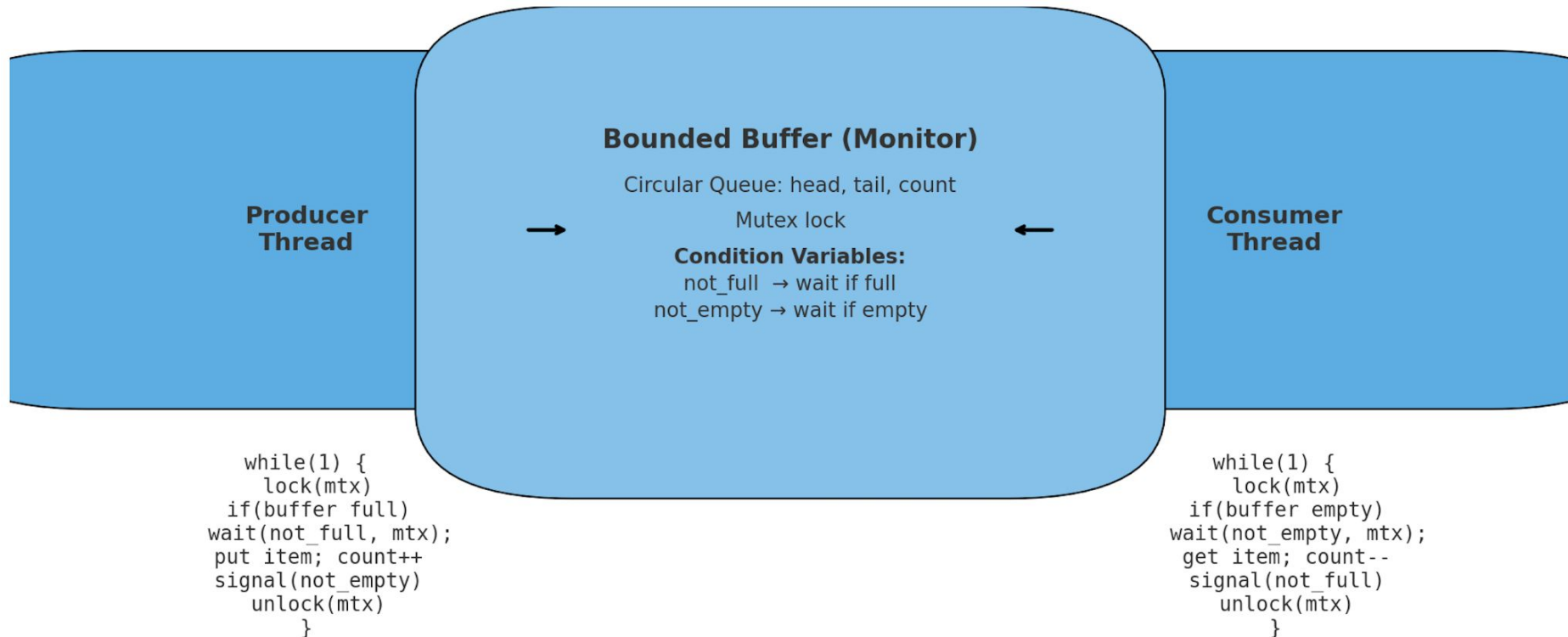
- Mutex Locks
- Condition variables
- Semaphores



Combine these two for
the Monitor approach



Monitor Approach



Note: wait(cond, mtx) releases the mutex and suspends the thread until signaled. This is NOT busy waiting — the thread sleeps and consumes no CPU while waiting.



C Thread Safety Demo

- Download it here!

